REMARKS

Applicant thanks Examiner for the detailed review of the application.

Claim Status

Claim 26 has been amended.

Claims 5, 8, and 32 remain cancelled.

Claim Rejections -35 USC § 102(b)

The Office Action has rejected Claims 1-4, 6-7, 9-31, and 33-41, under 35 U.S.C. § 102(b) as being anticipated by U.S. Pub. No.2001/0037434 to Hughes et al. (referred to hereinafter as "Hughes"). However, the Office Action has failed to make a prima facie case of anticipation for the claims, and such, the applicant respectfully submits that the rejections should be withdrawn.

"[F]or anticipation under 35 U.S.C. 102, the reference must teach *every aspect* of the claimed invention ..." MPEP 706.02 (emphasis added). "The identical invention must be shown *in as complete detail as contained in the ... claim.*" *Richardson v., Suzuki Motor Co.*, 868 F. 2d 1226, 1236, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989) (emphasis added).

Hughes discloses forwarding from in flight stores that do not have data yet available. For example, in the abstract and summary (para 0010), Hughes discusses creating an entry for a load that is to load from a store which does not have data available in a dependency link file. The dependency link file monitors result data for data from the store, i.e. store data. When the store data is detected, the store data is forwarded to the load to decrease load latency.

Therefore, generally speaking, Hughes only discloses either directly forwarding store data from a reorder buffer that includes the store data to a load or waiting for the store data to be obtained through execution and then directly forwarding the store data to the load. Note two significant items: (1) Hughes does not disclose blocking of store data from being forwarded to

another instruction, such as a load or a non-speculative load, or blocking store data from being stored in memory; and (2) Hughes does not delineate forwarding or blocking between instructions being speculative or non-speculative.

Applicant's claim 1 includes "blocker logic to **prevent data associated with a store instruction of the speculative thread from being forwarded to an instruction of a non-speculative thread**," (emphasis added). In fact, Hughes only discusses a prior art method of forwarding data from in-flight stores, i.e. waiting for a result of store data to be obtained to immediately forward it to a load to reduce load latency. However, no where does Hughes discuss blocking of the store data. The Office Action cites paragraph 0076, which states that "certain load instruction may be restricted to be performed non-speculatively." Here, Hughes is stating that loads may be restricted to non-speculative performance, not blocking of forwarding of store data to an instruction of a non-speculative thread. For example, if a speculative store operation is performed out-of-order before a restricted non-speculative load operation, then when the load is performed, non-speculatively, the speculative store is held in a store or re-order buffer. Here, the speculative store data, whether retrieved from the store buffer or waited for due to the data being unavailable, may still be provided to the non-speculative load operation, since the speculative store exists in the store buffer with the non-speculative store is performed. In other words, restricting a load operation to be performed non-speculatively, i.e. not during a predicted branch of execution but rather during a current program branch/flow, is not the same as blocking store data associated with a speculative store from being forwarded to a non-speculative instruction.

Furthermore, Hughes does not disclose "marking logic **to mark instruction information** for an instruction of a speculative thread **as speculative**." The Office Action points to paragraph 0082, which describes instructions may be speculatively fetched. However, it is clear that speculatively

fetching an instruction and marking instruction information as speculative are clearly different. Here, Hughes only describes speculatively fetching instruction in a branch. No marking is necessary, because if a branch is deemed non taken, then the buffer from that point on is flushed. In contrast, applicant, in claim 1, includes marking logic to mark instruction information…as speculative, not just the fetching of speculative instructions, as in Hughes.

Applicant's claim 11 includes a similar element of, "dependence blocker logic to prevent data associated with a store instruction of a speculative thread from being forwarded to an instruction of a non-speculative thread." Note from above, that Hughes only discloses restriction of some loads to non-speculative, not blocking data associated with a speculative thread store instruction from being forwarded to a non-speculative thread instruction. The Office Action points to paragraph 0061 of Hughes, which discloses "having storage locations for data and addresses information for pending loads or stores which have not accessed data cache **28**.." However, as noted above, this buffer is purely to hold stores, which have not received store data yet, in comparison to the other buffer that includes stores matched with stored data. Normal out-of-order execution includes some store operations in a buffer that have not yet received associated store data, not from blocking, but rather through allocation of the store operation in front-end execution of a pipeline and obtaining of store data during later pipeline execution.

Therefore, as described in Hughes, when a load hits the first pending buffer, the dependence monitor waits until the store data is obtained and immediately forwards it to the load. Note that, here, there is no logic described to block store data. In fact, the store data, through normal pipeline execution, has not been obtained yet through execution to be matched with the store instruction held in the first buffer. As soon as the result is obtained it is forwarded the pending load. In addition, note that Hughes does not describe forwarding or not forwarding the store data, once obtained. To

the load based on whether it is associated with a non-speculative thread, as in applicant's claim 11. Furthermore, reorder buffer tags, as described in paragraph 0056 Hughes, a purely to identify the set and offset within the set a particular instruction corresponds to, which is to say the reorder buffer tag identifies a location, not whether the instruction is speculative or non-speculative.

Applicant's claim 21 includes the elements, "determining if **the load instruction and the in-flight store instruction each originate with a speculative thread**... forwarding, if the dependence check is successful, store data... declining to forward, if the dependence check is not successful, the store data ." Note that Hughes does not disclose anywhere determining of both the load instruction and the in-flight store originate with a speculative thread. Furthermore, Hughes does not disclose declining to forward store data if the dependence check is not successful. At paragraph 0082 in regards to SMC checks, which is also discussed in reference to Figure 20, Hughes describes a prior art check to determine if a current store is to update in-flight stores in a processor, where if it is determined that in flight instructions have been updated, corrective action is taken (e.g. discarding the instructions subsequent...). Note there is no determination of whether a load or store is speculative or not. Furthermore, there is no declining to forward store data, but rather only discarding instructions subsequent. Hughes only discloses committing stores [to data cache 28] when they become non speculative. However, this is stating only that store data is not update in data cache 28 until they become non-speculative, not that the data held in the buffer before commitment is blocked from being forwarded to non-speculative thread instructions

As a simplified example of Hughes, assume an address A in a data cache 28 holds a current value of one. When a load from address A is detected, a store buffer or re-order buffer may be searched, to see if a current store operation is to update address A, i.e. an in-flight store operation is to update address A. If a store operation exists in the buffer with store data, such as a value of two,

the value of two is forwarded to the load instead of the data cache value of one, as the value of two is the valid current data for address A. If no store data is available but the store operation exists, then the dependence monitor waits for the store data to be provided. For example, assume the store data is a result of an addition of one plus one waiting to be performed. As soon as the result is obtained, the new value of two is forwarded to the load. That new value of two is not committed, i.e. updated to data cache 28, until the store is retired. Note two significant items from Hughes disclosure: (1) there is no description of blocking store data, but rather only determining if a store exists in a buffer and then either immediately forwarding that data to a load or waiting for the resultant store data to be obtained from execution; and (2) there is no distinction in Hughes between the store or the load being speculative, being associated with a speculative thread, being non-speculative, or being associated with a non-speculative thread.

Applicant's claim 26 includes, "determining a cache line in a cache corresponding to a speculative thread cache write request includes dirty non-speculative data." As stated above, Hughes does not discuss distinguishing or determining whether data or instructions are speculative. In addition, Figure 24 and the corresponding text, describes exponential back off for obtaining ownership of a cache line between processors. Note, that the discussion does not describe determining if data is non-speculative dirty data or generating a write-back in response to such.

Applicant's claim 34 includes, "a storage area to include a speculation identifier (ID) field, the speculation ID field to hold a first value to indicate an associated store instruction is associated with the speculative thread." Hughes paragraph 56, as stated above, only describes reorder buffer tags that for particular instructions reference corresponding locations, i.e. a set and offset within the set, not to indicate an associated instruction is associated with a speculative thread as in claim 34. Furthermore, as described above, Hughes does not disclose control logic to prevent data associated

with the store instruction from being consumed by the non-speculative thread, as in applicants claim 34. As illustrated above, load instructions being performed non-speculatively and stores committing to a data cache, do not describe preventing data **of a speculative thread** from being consumed by **the non-speculative thread.**

As a result, applicant respectfully submits that independent claims 1, 11, 21, 26, and 34, as well as their dependent claims, are now in condition for allowance for at least the reasons stated above.

If there are any additional charges, please charge Deposit Account No. 50-0221. If a telephone interview would in any way expedite the prosecution of the present application, the Examiner is invited to contact David P. McAbee at (503) 712-4988.

Respectfully submitted,
Intel Corporation

Dated: January 24, 2008

/s/David P. McAbee/Reg. No. 58,104/
David P. McAbee
Reg. No. 58,104

Intel Corporation
M/S JF3-147
2111 NE 25th Avenue
Hillsboro, OR 97124
Tele – 503-712-4988
Fax – 503-264-1729